Simple Movement and Detection in Discrete-Event Simulation

This note will discuss the way simple motion (linear, uniform, two-dimensional) and simple sensing ("cookie-cutter") is modeled in Discrete-Event Simulation (DES). Recall that in Discrete-Event Simulation, time does not advance in regular intervals or steps. Rather, the simulated time is moved to the time of the next occuring Event which is then processed — first its state is changed, then event cancelations, if any, performed, and finally scheduled events, if any, are scheduled. In between the occurrence of events, there is no change in the value of any state variable.

Modeling movement presents a difficulty to the Discrete-Event approach, since the state of an entity in motion (its location, for instance) is in constant change when an entity is in motion. This difficulty is overcome by the notion of *implicit state*. An implicit state is one that is not explicitly stored in state variables (instance variables in an object-oriented framework) but rather implicitly determined from other state variables. An entity that moves in uniform, linear motion may have its position modeled by implicit state in that its position is not stored as an instance variable but is computed "on demand," as described below. The implicit state of position is determined from three explicit state variables: the entity's position when it started the move, the time it started the move, and its velocity.

We will consider the simplest kind of sensing, the "cookie-cutter." A cookie-cutter sensor sees everything that is within its range R, and must be notified at the precise time a target enters it range. In a time-step simulation, cookie-cutter detection is very easy. Simply compute the distance between the sensor and the target at each time step. If the target is within distance R of the sensor, then a detection occurs. The precise time of detection cannot be determined, of course, but only up to the depends on the length of the time step chosen.

1 Simple Movement

The simplest possible movement is uniform, linear motion. A moving entity starts its move at some initial position x_0 at time t_0 and begins moving with velocity $\vec{v}$. Thus, the location of the entity at time t is $x_0 + (t - t_0)\vec{v}$. Equivalently, the location of the entity s time units after it began its movement is $x_0 + s\vec{v}$.

In a Discrete Event Simulation model the location of moving entities is modeled using implicit state, rather than explicit state, as mentioned above. Rather than storing the current location of the entity at all times, enough information is stored so that the current position can be computed easily whenever desired using "dead reckoning." For uniform linear motion, it is enough to store: (1) The initial position x_0 (i.e. the location of the entity just prior to when it started moving); (2) The velocity vector $\vec{v}$; and (3) The time it started moving t_0. Then the equations of motion of the previous paragraph are applied whenever an entity "asks" the entity its position.

The coordinates and velocities of the entities are all in some common base coordinate system, so the motion represented above can be considered *absolute* motion in the base coordinates. Often it is desirable to consider location and motion relative to some entity's coordinates. In that case, the locations and velocities can be represented relative to that entity's coordinates. For most purposes the entities' coordinate systems may be considered to be simply a translation of the base coordinate

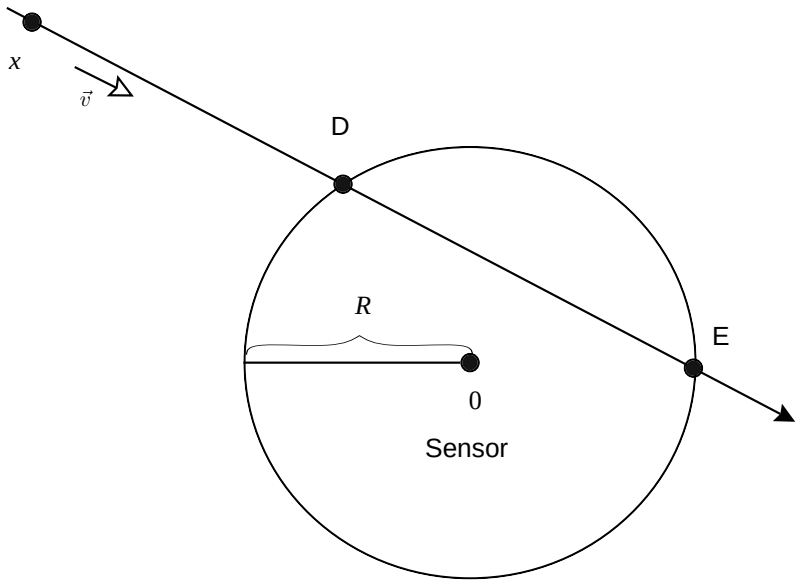

Figure 1: Cookie-Cutter Detection: The Basic Scenario

system.[1] Thus, an entity at position y in base coordinates is at position $y - x$ in the coordinates of an entity located at position x (base coordinates). Relative velocity is equally simple for uniform linear motion. Suppose the equations of motion for two entities are given by $x_i + t\vec{v}_i$ for $i = 1, 2$. Then in the coordinate system of entity 1, the motion of entity 2 is given by $(x_2 - x_1) + t(\vec{v}_2 - \vec{v}_1)$. Thus, relative to the first entity, the motion of the second is uniform and linear with starting position $x_2 - x_1$ and velocity $\vec{v}_2 - \vec{v}_1$.

2 Simple Detection

The simplest kind of detection is the cookie-cutter sensor. Consider the simplest scenario, with one sensor and one moving target. We will consider the sensor to be located at the origin in a two-dimensional coordinate system. At time 0 the target starts at point x (relative to the sensor) and proceeds with constant velocity $\vec{v}$ (again, relative to the sensor). This situation is illustrated in Figure 1. The problem is to determine the time t_d at which the target enters the sensor's range. Note that the location of the target at the time of detection is given by $x + t_d\vec{v}$. Since detection occurs whenever the distance between the target and the sensor is exactly R, the time at which this occurs is the solution to the equation

$$||x + t\vec{v}|| = R. \tag{1}$$

Equivalently, t_d is the solution to

$$||x||^2 + 2tx \cdot \vec{v} + t^2||v||^2 = R^2, \tag{2}$$

[1] Rotations can also be considered, but the computations are complicated substantially. Handling rotations is left as an exercise for the interested reader.

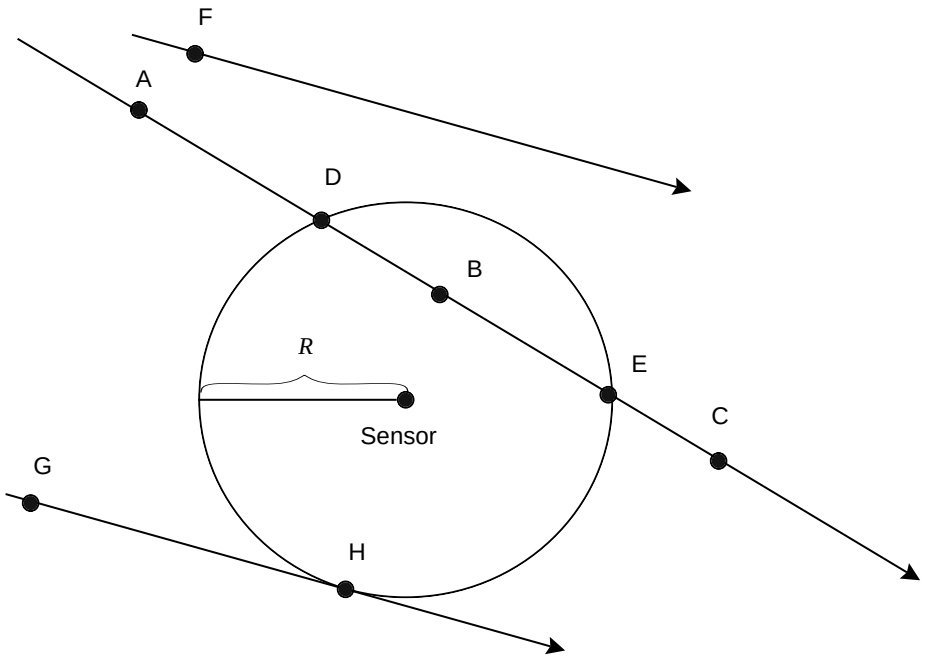

Figure 2: Cookie-Cutter Detection: All Possibilities

where "·" represents the vector inner product and || || is the length of a vector. Equation (2) is a quadratic in t, so the solutions are given by

$$t = -\frac{x \cdot \vec{v}}{||v||^2} \pm \frac{\sqrt{||v||^2(R^2 - ||x||^2) + (x \cdot \vec{v})^2}}{||v||^2}, \tag{3}$$

providing the expression under the radical is non-negative. If the target starts out of range but is eventually detected, then the solutions in 3 are both real and positive. This is the situation depicted in Figure1. In this case, the smaller of the two solutions is t_d, the time of detection, and the larger solution t_e is the exit time, the time when the target leaves the sensor's range. At time t_d therefore the target will be at location D and at time t_e it will be at location E.

The expressions in Equation 2 can now be used to schedule the detection of the target as well as the target's exit time. The above calculations assumed that the starting time was 0.0. In general, the interpretation of t_d and t_e would be the amount of time elapsed *after* the target started moving. This fits nicely with DES, since Events are scheduled after a time delay and, in general, relative times are easier to model than absolute times.

The possible outcomes may be summarized as follows:

- *Both roots positive.* The sensor's range will be entered after a delay of the smaller root and exited after a delay of the larger root. In Figure 2, this corresponds to a target starting at point A heading through C. The sensor's range is entered at point D and exited at point E. In case of equality, the target will be on a tangent to the range ring. In Figure 2, the target starts at point G and the tangent occurs at point H.

- *One positive and one negative root.* The target is already within the sensor's range and will exit after a delay of the positive root. In Figure 2, the target starts at B and proceeds through C. The sensor's range is exited at point E.
- *Both roots negative.* The target is outside the sensor's range and is moving away from the sensor. The target will never enter the sensor's range. In Figure 2, the target starts at point C and heads away from the sensor.
- *No real roots.* The target will never enter the sensor range. In Figure 2, the target starts at point F and proceeds in the direction indicated, missing the sensor's ring.

All the above cases assume, of course, that the target will not stop, change direction or change speed. In any of these cases, the results must be recomputed. Any events that had been scheduled based on the original computations are of course invalid and must be canceled and new events scheduled, if necessary.

Although exceedingly simple, the cookie-cutter sensor is useful for simulating more sophisticated sensors. For example, the sensor's maximum range could be the "cookie" so that the detection algorithm is simply triggered when a target enters the range. The timing of the entry and exit would be determined by the cookie-cutter algorithm described in this note. For example, suppose that the time to detect a target after it enters a sensor's range is exponentially distributed with mean μ. This sensor could be simulated by scheduling the actual detection with an exponential delay following the event that the target enters the range. If the target exits the sensor's range before that time has elapsed, then that detection must be canceled, of course.